Computers & Operations Research 39 (2012) 2612–2624

Contents lists available at SciVerse ScienceDirect

Computers & Operations Research

journal homepage: www.elsevier.com/locate/caor

A hierarchical solution approach for a multicommodity distribution problem under a special cost structure ☆

Esra Koca [a], E. Alper Yıldırım [b,*]

[a] Department of Industrial Engineering, Bilkent University, 06800 Bilkent, Ankara, Turkey
[b] Department of Industrial Engineering, Koç University, 34450 Sarıyer, Istanbul, Turkey

ARTICLE INFO

Available online 28 January 2012

Keywords:
Multicommodity distribution
Logistics
Network design
Demand aggregation
Hierarchical Approach

ABSTRACT

Motivated by the spare parts distribution system of a major automotive manufacturer in Turkey, we consider a multicommodity distribution problem from a central depot to a number of geographically dispersed demand points. The distribution of the items is carried out by a set of identical vehicles. The demand of each demand point can be satisfied by several vehicles and a single vehicle is allowed to serve multiple demand points. For a given vehicle, the cost structure is dictated by the farthest demand point from the depot among all demand points served by that vehicle. The objective is to satisfy the demand of each demand point with the minimum total distribution cost. We present a novel integer linear programming formulation of the problem as a variant of the network design problem. The resulting optimization problem becomes computationally infeasible for real-life problems due to the large number of integer variables. In an attempt to circumvent this disadvantage of using the direct formulation especially for larger problems, we propose a Hierarchical Approach that is aimed at solving the problem in two stages using partial demand aggregation followed by a disaggregation scheme. We study the properties of the solution returned by the Hierarchical Approach. We perform computational studies on a data set adapted from a major automotive manufacturer in Turkey. Our results reveal that the Hierarchical Approach significantly outperforms the direct formulation approach in terms of both the running time and the quality of the resulting solution especially on large instances.

1. Introduction

The problem considered in this paper was motivated by the need to implement an improved and more cost-effective spare parts distribution system for a major automotive manufacturer in Turkey. The company has a central depot that houses a number of spare parts, henceforth referred to as *items*; there are a number of retailers, henceforth referred to as *demand points*, geographically dispersed all around Turkey, that are served by this depot. The demand points place their orders with the depot on a daily basis. The distribution of the items is performed by a set of identical vehicles. The demand of each demand point can be satisfied by different vehicles and a vehicle is allowed to serve multiple demand points. The objective is to satisfy the demand of each demand point with the minimum total distribution cost while respecting the capacity of each vehicle. Clearly, this setting encompasses a vast majority of distribution problems in real life.

Typically, such distribution problems are formulated as an instance of the Vehicle Routing Problem (VRP), which asks for the minimum-cost solution of a distribution problem from a central depot to a number of demand points. In the VRP, the transportation cost is assumed to depend linearly on the distance traveled by each vehicle.

A variant of the VRP which is more closely related to the problem under consideration in this paper is the Split Delivery Vehicle Routing Problem (SDVRP) (see, e.g., [9,10,6,2,3]). Similar to our problem, the SDVRP allows the delivery to a demand point to be split among two or more vehicles. It has been shown that allowing split deliveries may yield savings in terms of both the total cost and the number of vehicles required (see, e.g., [1]).

The main difference between the distribution problem considered in this paper and the SDVRP is the cost structure. More precisely, for a given vehicle, the cost structure in our problem is dictated by the farthest demand point from the depot among all demand points served by that vehicle. The farthest demand point determines the "main route" of a vehicle, which is simply given by the shortest path from the depot to that demand point. The main route of a vehicle incurs a fixed cost, which is primarily

☆ This work is supported in part by TÜBİTAK (Turkish Scientific and Technological Research Council) Grant 109M149 and by TÜBA-GEBİP (Turkish Academy of Sciences Young Scientists Award Program).

* Corresponding author.
E-mail addresses: ekoca@bilkent.edu.tr (E. Koca), alperyildirim@ku.edu.tr (E.A. Yıldırım).

0305-0548/$ - see front matter © 2012 Elsevier Ltd. All rights reserved.
doi:10.1016/j.cor.2012.01.007

related to the length of the route. If a vehicle is used to satisfy the (partial) demand of a demand point other than the farthest demand point, an additional cost is charged, which usually reflects the overhead incurred by such a visit (e.g., the additional distance traveled by the vehicle as a result of the detour from its main route and the additional time due to stopover and unloading). The problem studied in this paper boils down to the determination of the number of vehicles to be used in transportation, the assignment of demand points to each vehicle, and the allocation of the capacity of each vehicle to the demand points to be served by that vehicle. The objective is to minimize the total transportation cost given by the sum of transportation costs of individual vehicles computed with respect to the aforementioned cost structure. To the best of our knowledge, such a cost structure has not previously been studied in the distribution literature. As such, the problem under consideration is quite different from a typical vehicle routing problem and requires a specific solution approach. We refer to this problem as the Multicommodity Distribution Problem with Fixed Assignment Costs (MDPFAC).

Let us briefly discuss the rationale behind this cost structure. The physical distribution operations of the aforementioned automotive manufacturer in Turkey are carried out by a separate transportation company. However, the manufacturer is fully in charge of deciding how to serve the demand of each demand point. The transportation company is responsible for supplying vehicles and charges the manufacturer based on the assignment of demand points to each vehicle. The adoption of such a cost structure leads to simpler routing decisions since the main route of a vehicle is simply given by the shortest path from the depot to the farthest demand point to be served by that vehicle. If, in addition, a vehicle serves any other demand point, the vehicle simply needs to follow a path that would minimize the deviation from its main route. This simplicity is an attractive feature of this particular cost structure from the point of view of the transportation company. From the manufacturer's perspective, such a cost structure enables the manufacturer to design simpler contracts with the transportation company. Indeed, since the main route of a vehicle is determined by the farthest demand point served by that vehicle, it suffices to define overhead costs for only the remaining demand points that are closer to the depot than the farthest demand point. This cost structure can therefore lead to a considerable decrease in terms of data requirements especially in comparison with a VRP instance of similar size, where one usually needs the distance information between all pairs of demand points.

In this paper, we study the MDPFAC from the manufacturer's point of view. Given this cost structure, we aim to satisfy the demand of each demand point with the minimum total distribution cost. We develop a novel integer linear programming formulation of this problem as a variant of the network design problem. We first attempt to compute a solution by directly solving the resulting optimization model. However, due to the large number of integer and binary variables in this formulation, the optimization model becomes too difficult to solve to optimality even for medium-scale instances. We therefore propose a Hierarchical Approach in an attempt to effectively find a good solution in two stages. In the first stage, we apply a partial demand aggregation scheme in an attempt to reduce the size of the original problem. We solve the MDPFAC using the resulting partially aggregated demand information for each demand point, which constitutes the first stage. Due to the potentially dramatic reduction in the size of the problem, the resulting optimization model can usually be solved to optimality in reasonable time. The solution of the first stage determines the set of vehicles to be used along with their main routes, the set of demand points each vehicle will serve, and the total capacity allocated to each demand point in each vehicle. The second stage is aimed at disaggregating the demand of each demand point while respecting the solution of the first stage as closely as possible. One of the main computational advantages of the second stage is that the disaggregation scheme decomposes naturally into solving a number of smaller optimization problems, each of which can be much more effectively solved due to the considerably smaller number of demand points. Our computational results on a data set adapted from the aforementioned automotive manufacturer in Turkey reveal that the Hierarchical Approach significantly outperforms the direct formulation approach in terms of both the running time and the quality of the solution.

In our Hierarchical Approach, the second stage is aimed at computing a feasible solution of the original problem that respects the solution obtained from the first stage. However, since indivisibility of items in the aggregated part of the demand is completely ignored in the first stage, it may not always be possible to convert the solution from the first stage into a feasible solution of the MDPFAC. We therefore design the second stage in an attempt to minimize a natural measure of infeasibility with respect to the original problem. We identify an interesting connection between the second stage problem and the makespan scheduling problem on unrelated parallel machines, which allows us to establish a tight upper bound on the aforementioned infeasibility measure. This connection enables us to propose a feasibility restoration scheme to recover a feasible solution whose total cost is within a factor of two of the optimal total cost if the solution computed at the end of the second stage happens to be infeasible for the original problem.

The rest of this paper is organized as follows. We present a direct optimization formulation and our Hierarchical Approach in Section 2. Section 3 is devoted to the analysis of various properties of the solution computed by the Hierarchical Approach. We present computational results on a data set adapted from an automotive manufacturer in Turkey in Section 4. Finally, Section 5 concludes the paper with some future research directions.

2. Direct formulation and Hierarchical Approach

In this section, we give a formal definition of the MDPFAC considered in this paper. In an attempt to directly solve the problem, we develop an integer linear programming formulation. Due to the large number of integer and binary variables in the resulting optimization model, we propose a Hierarchical Approach, which attempts to solve the problem in two stages by first applying a partial demand aggregation scheme in the first stage followed by disaggregation of demand in the second stage.

We assume that there is a central depot which houses m different item types. There are n geographically dispersed demand points, each of which places orders for these items with the depot on a daily basis. The items are distributed to the demand points by a set $\mathcal{V}$ of identical vehicles. We assume that there is no restriction on the number of available vehicles. A vehicle is allowed to serve multiple demand points. Similarly, the demand of a demand point can be satisfied by different vehicles. However, we do assume that each unit of each item type is indivisible.

The cost of using a vehicle is dictated by the set of demand points served by that vehicle. There are two types of transportation costs. The use of a vehicle in transportation incurs a fixed cost $f_j, j=1,\ldots,n$, where j denotes the farthest demand point from the depot among all demand points served by that vehicle. Since the farthest demand point determines the main route of a vehicle, the cost f_j is typically related to the length of the shortest path from the depot to demand point j. We say that a vehicle belongs to class j if demand point j is the farthest demand point from the

depot among all demand points served by that vehicle, $j=1,\ldots,n$. Note that each vehicle used in transportation belongs to exactly one class. In addition, if a vehicle from class j is used to satisfy the (partial) demand of demand point k, where $k\neq j$, an additional cost of o_{jk} is charged independently of the amount delivered to demand point k. This "overhead" cost usually reflects the length of the detour from the main route of that vehicle and the additional time due to stopover and unloading. Note that a vehicle from class j is not allowed to serve demand points which are farther from the depot than demand point j. Therefore, for each vehicle from class j, $j=1,\ldots,n$, the overhead parameters o_{jk} need to be defined only for demand points k satisfying $f_k\leq f_j$. It follows that it is economically feasible to charge an overhead cost o_{jk} for a vehicle from class j only for demand points given by

$$\mathcal{O}_j := \{k\in\{1,\ldots,n\}:\ o_{jk}\leq f_k\leq f_j,\ k\neq j\},\quad j=1,\ldots,n. \tag{1}$$

The set $\mathcal{O}_j$ can be viewed as the set of demand points which are within a certain proximity of the shortest path from the depot to demand point j. For each vehicle, the transportation cost is given by the sum of the fixed cost and individual overhead costs. The total transportation cost is the sum of the transportation costs of the individual vehicles.

We denote by d_{ik} the number of units of item type i ordered by demand point k, $i=1,\ldots,m$; $k=1,\ldots,n$. For item type i, $w_i\in(0,1]$ denotes the ratio of the capacity of a vehicle occupied by one unit of that item, $i=1,\ldots,m$. Note that w_i can be viewed as the inverse of the largest number of units of item type i that can fit into a vehicle. We make the standard assumption that any combination of items can be loaded on a vehicle as long as the sum of the corresponding values w_i does not exceed the capacity of the vehicle, which is assumed to be one. While this assumption ignores the other packing restrictions that may result from different geometries of the item types, this drawback can be circumvented, if necessary, by replacing each item type by a regular enclosing object such as a rectangular box and inflating the values of w_i accordingly.

The MDPFAC jointly asks for the number of vehicles from each class to be used to meet the demand, the determination of the set of demand points that will be served by each vehicle, and the allocation of the capacity of each vehicle to the demand points served by that vehicle. The main objective is to satisfy the demand of each demand point with the minimum total transportation cost while respecting the capacity of each vehicle.

We summarize the parameters of the problem for future reference:

- m: the number of different item types;
- n: the number of demand points;
- $\mathcal{V}$: the set of vehicles with identical capacities;
- f_j: the fixed cost incurred for the use of a vehicle from class j, $j=1,\ldots,n$;
- o_{jk}: the overhead cost incurred for a vehicle from class j due to serving demand point k, $j=1,\ldots,n$; $k\in\mathcal{O}_j$;
- d_{ik}: the number of units of item type i ordered by demand point k, $i=1,\ldots,m$; $k=1,\ldots,n$;
- w_i: the ratio of the capacity of a vehicle occupied by one unit of item type i, $i=1,\ldots,m$.

2.1. Direct formulation

In this section, we develop an integer linear programming formulation of the MDPFAC. Note that the fixed cost and the overhead costs of a vehicle are dictated by its class, which, in turn, is determined by the farthest demand point from the depot among all demand points served by that vehicle. In our optimization model, for each class, we will define a set of potential vehicles that are available to be used in transportation. However, since we do not know a priori the number of vehicles from each class that will be used in an optimal solution, we first propose a procedure that enables us to define a sufficiently large number of potential vehicles for each class in our model.

Since each vehicle from class j is required to deliver at least one unit to demand point j, $j=1,\ldots,n$, it is clear that the number of vehicles from class j cannot be larger than the total number of units to be delivered to demand point j due to the indivisibility of items. This is our first upper bound on the number of vehicles from class j. However, this upper bound can be quite weak especially if there is a large number of small items ordered by demand point j. Since the number of potential vehicles directly affects the size of the optimization model, it is important to obtain a sharper upper bound if the aforementioned bound happens to be weak. We therefore devise a simple procedure to obtain another upper bound on the number of potential vehicles from each vehicle class.

Clearly, in an optimal solution, a vehicle from class j can only serve demand points in $\mathcal{S}_j$, where

$$\mathcal{S}_j := \{j\}\cup\mathcal{O}_j,\quad j=1,\ldots,n. \tag{2}$$

Therefore, in the extreme case, the total demand of all demand points in $\mathcal{S}_j$ can be satisfied using only vehicles from class j. Regarding this as a bin packing problem in which each vehicle is identified with a bin, we apply the following procedure, called *Procedure UB*, to determine an upper bound on the number of potential vehicles.

Procedure UB starts with demand point j. We sort the items ordered by demand point j in the order of nonincreasing item sizes w_i. We use the well-known first-fit algorithm: we first open a bin of capacity one. We start packing items in the given order one by one and open the second bin only when the next item no longer fits into the first bin. For each of the remaining items, we try packing it into the smallest numbered bin whose residual capacity is large enough to accommodate the item. We open a new bin only when the item does not fit into any one of the previously opened bins. We continue this procedure until all the items are packed into the bins. Let b_j denote the number of bins computed by this algorithm.

Next, Procedure UB considers each demand point $k\in\mathcal{O}_j$ one by one. In the MDPFAC, each vehicle from class j should deliver at least one item to demand point j. Our aim is to construct a feasible solution that satisfies all the demand of demand point k using only vehicles from class j. Therefore, we apply a variant of the first-fit algorithm that takes into account the delivery requirement to demand point j. We similarly sort the items ordered by demand point k in the order of nonincreasing item sizes w_i. We also assume that *all* the items ordered by demand point j are initially available to be packed into bins. We apply the following variant of the first-fit algorithm: each time a new bin should be opened by the algorithm, we first pack the *largest* unpacked item ordered by demand point j into the new bin and then we continue to pack the items ordered by demand point k into the open bins using exactly the first-fit algorithm. This procedure ensures that each bin contains exactly one item ordered by demand point j. We continue in this fashion until we pack all the items ordered by demand point k or until we fail to open a new bin since we run out of items ordered by demand point j. In the former case, we denote by b_{jk} the number of bins computed by this procedure. In the latter case, we define $b_{jk}=+\infty$.

We repeat the same procedure for each demand point $k\in\mathcal{O}_j$ and compute the corresponding number of bins b_{jk}. It is worth noticing that all the items ordered by demand point j are initially assumed to be available to be packed before we start each

demand point $k \in \mathcal{O}_j$. This assumption is crucial in the proof of the following result.

Theorem 2.1. *For a given instance of the MDPFAC, the number of vehicles from class j that can be used in any optimal solution is bounded above by*

$$u_j := \min\left\{ b_j + \sum_{k \in \mathcal{O}_j} b_{jk}, \sum_{i=1}^{m} d_{ij} \right\}, \quad j = 1, \ldots, n, \tag{3}$$

where b_j and b_{jk} are obtained by using Procedure UB.

Proof. The second argument on the right hand side of (3) is simply the total number of units to be delivered to demand point j. Therefore, it suffices to prove the case in which u_j is determined by the first argument on the right hand side of (3).

Suppose, for a contradiction, that there exists an optimal solution, say Solution 1, that uses more than $u_j = b_j + \sum_{k \in \mathcal{O}_j} b_{jk}$ vehicles from class j. Note that vehicles from class j can serve only the demand of demand points in $\mathcal{S}_j$. We will construct a strictly better solution that uses fewer vehicles from class j.

First, we construct a partial solution, which considers only vehicles from class j, using the following procedure, called *Procedure FS*. We consider only the items that are delivered to demand points in $\mathcal{S}_j$ using vehicles from class j in Solution 1. For each demand point in $\mathcal{S}_j$, we separately sort the items in nonincreasing item sizes. Originally all such items are available to be packed into the bins. We consider each demand point $k \in \mathcal{O}_j$ one by one. Similar to Procedure UB, we start packing items of demand point k one by one into bins of capacity one using the first-fit algorithm, packing first the largest unpacked item ordered by demand point j every time we open a new bin. Before we start the next demand point $k' \in \mathcal{O}_j$, we "close" all the open bins, open a new bin, and continue packing an item ordered by demand point j every time we open a new bin. In contrast to Procedure UB, upon starting the next demand point, we are allowed to use the largest unpacked item only from the set of *remaining* items ordered by demand point j. After we pack all the demand of each demand point $k \in \mathcal{O}_j$ in this manner, we close all open bins. If there are still remaining items to be delivered to demand point j, we start a new bin and pack these items using the usual first-fit algorithm.

We claim that Procedure FS uses at most u_j vehicles from class j. For each demand point $k \in \mathcal{O}_j$, Procedure UB considers all the demand whereas Procedure FS considers only the subset of the demand delivered by vehicles from class j in Solution 1. In addition, each time a new bin is opened while packing the items to be delivered to demand point k, Procedure FS never packs a larger item ordered by demand point j compared to the item packed by Procedure UB. Therefore, the residual capacity in each newly opened bin using Procedure FS is at least as large as that in its counterpart in Procedure UB. It follows that Procedure FS uses at most b_{jk} vehicles for each demand point $k \in \mathcal{O}_j$. By a similar argument, the remaining items ordered by demand point j after finishing each demand point $k \in \mathcal{O}_j$ can be packed into at most b_j bins. Therefore, at most $u_j = b_j + \sum_{k \in \mathcal{O}_j} b_{jk}$ vehicles from class j are used by Procedure FS. Note that, by our assumption, the number of vehicles from class j used by Solution 1 is strictly larger than u_j, which constitutes a lower bound on the number of units delivered to demand point j by vehicles from class j in Solution 1. Therefore, during Procedure FS, we can never run out of items ordered by demand point j whenever we should open a new bin. It follows that the partial solution computed by Procedure FS, denoted by Solution 2, satisfies the vehicle capacity constraints and delivery requirements to demand point j.

The total cost of the vehicles from class j used by Solution 2, denoted by v_j, satisfies

$$v_j \le f_j u_j + \sum_{k \in \mathcal{O}_j} o_{jk} b_{jk}.$$

Let $\overline{u}_j > u_j$ denote the total number of vehicles from class j and let $\overline{b}_{jk}$ denote the number of vehicles from class j that deliver to demand point $k \in \mathcal{O}_j$ in Solution 1. Therefore, the total cost of vehicles from class j in Solution 1 is given by

$$\overline{v}_j = f_j \overline{u}_j + \sum_{k \in \mathcal{O}_j} o_{jk} \overline{b}_{jk}. \tag{4}$$

Note that $\overline{v}_j \le v_j$ since, otherwise, we can strictly improve the solution by replacing the part of Solution 1 that uses vehicles from class j by Solution 2, which contradicts the optimality of Solution 1. Since $\overline{u}_j > u_j$ by our assumption, the total overhead cost of vehicles from class j in Solution 1 should be strictly smaller than that of Solution 2. It follows that

$$\mathcal{D}_j := \{k \in \mathcal{O}_j : \overline{b}_{jk} < b_{jk}\} \neq \emptyset. \tag{5}$$

We can now construct a hybrid feasible solution as follows. In Solution 1, consider the vehicles from class j that deliver to demand points in $\mathcal{D}_j$. In each such vehicle, we keep all the items delivered to demand points in $\mathcal{D}_j$ in addition to the items delivered to demand point j. By (5), the number of such vehicles is strictly smaller than the corresponding number given by $\sum_{k \in \mathcal{D}_j} b_{jk}$ in Solution 2. From these vehicles, we then remove all the remaining items ordered by demand points not belonging to $\mathcal{D}_j$ (if any). Using these demand points and the remaining items to be delivered to demand point j, we apply Procedure FS to construct the part of the solution corresponding to demand points $k \notin \mathcal{D}_j$ as well as demand point j itself. By a similar argument, Procedure FS cannot open more bins than Procedure UB. Therefore, at most $b_j + \sum_{k \notin \mathcal{D}_j} b_{jk}$ new vehicles are introduced. Let Solution 3 denote this partial solution. For vehicle class j, it follows from (5) that the total overhead cost of Solution 3 is bounded above by that of Solution 1. On the other hand, the total fixed cost of Solution 3 is strictly smaller since the total number of vehicles is strictly smaller than $\overline{u}_j$. Therefore, replacing all vehicles from class j in Solution 1 by Solution 3 yields a strictly better feasible solution that uses at most u_j vehicles, which contradicts the optimality of Solution 1. □

By Theorem 2.1, for each vehicle class j, we define u_j potential vehicles in our model and label vehicles from class j using $1, 2, \ldots, u_j$. The decision variables are defined as follows:

- x_{ijkv}: the number of units of item type i delivered to demand point k by vehicle v from class j, $i = 1, \ldots, m$; $j = 1, \ldots, n$; $k \in \mathcal{S}_j$; $v = 1, \ldots, u_j$;
- $y_{jkv} = \begin{cases} 1 & \text{if vehicle } v \text{ from class } j \text{ serves demand point } k, \\ 0 & \text{otherwise,} \end{cases}$
  $j = 1, \ldots, n;\ k \in \mathcal{O}_j;\ v = 1, \ldots, u_j;$
- $z_{jv} = \begin{cases} 1 & \text{if vehicle } v \text{ from class } j \text{ is used,} \\ 0 & \text{otherwise,} \end{cases}$
  $j = 1, \ldots, n;\ v = 1, \ldots, u_j.$

The integer variables x_{ijkv} determine the allocation of the demand of each demand point for each item type among vehicles in each

class. The fixed costs and the overhead costs are accounted for by the binary variables y_{jkv} and z_{jv}, respectively. The MDPFAC admits the following integer linear programming formulation:

$$\min \quad \sum_{j=1}^{n}\sum_{v=1}^{u_j} f_j z_{jv} + \sum_{j=1}^{n}\sum_{k\in\mathcal{O}_j}\sum_{v=1}^{u_j} o_{jk} y_{jkv} \tag{6}$$

$$\text{s.t.} \sum_{j:k\in\mathcal{S}_j}\sum_{v=1}^{u_j} x_{ijkv} = d_{ik}, \quad i=1,\ldots,m;\ k=1,\ldots,n, \tag{7}$$

$$\sum_{i=1}^{m}\sum_{k\in\mathcal{S}_j} w_i x_{ijkv} \le z_{jv}, \quad j=1,\ldots,n;\ v=1,\ldots,u_j, \tag{8}$$

$$\text{(DA)} \quad \sum_{i=1}^{m} w_i x_{ijkv} \le y_{jkv}, \quad j=1,\ldots,n;\ k\in\mathcal{O}_j;\ v=1,\ldots,u_j, \tag{9}$$

$$\sum_{i=1}^{m} x_{ijjv} \ge z_{jv}, \quad j=1,\ldots,n;\ v=1,\ldots,u_j, \tag{10}$$

$$x_{ijkv} \ge 0, \quad i=1,\ldots,m;\ j=1,\ldots,n;\ k\in\mathcal{S}_j;\ v=1,\ldots,u_j, \tag{11}$$

$$x_{ijkv} \text{ integer}, \quad i=1,\ldots,m;\ j=1,\ldots,n;\ k\in\mathcal{S}_j;\ v=1,\ldots,u_j, \tag{12}$$

$$y_{jkv} \in \{0,1\}, \quad j=1,\ldots,n;\ k\in\mathcal{O}_j;\ v=1,\ldots,u_j, \tag{13}$$

$$z_{jv} \in \{0,1\}, \quad j=1,\ldots,n;\ v=1,\ldots,u_j. \tag{14}$$

The objective function (6) is given by the sum of individual fixed costs and overhead costs. The constraint set (7) ensures the satisfaction of the demand of each demand point for each item type using only vehicles from classes that are allowed to serve that demand point. The constraints (8) correspond to vehicle capacity restrictions by definition of w_i since each z_{jv} is a binary variable. If a vehicle serves a demand point different from the one at its final destination, the corresponding binary variable y_{jkv} is set to one by (9). The constraints (10) ensure that a vehicle from class j can be used in transportation if and only if it serves demand point j. Together with (7), the constraints (10) guarantee that demand point j is the farthest demand point from the depot among all demand points served by each vehicle from class j. The constraints (11) through (14) define the range of values each decision variable can take. We denote the optimization model by (DA).

We close this section by establishing a connection between the optimization model (DA) and a particular network design problem called the Capacitated Concentrator Location Problem (CCLP) on star/star networks. The CCLP is defined as follows: given a set of terminals with known demands and a set of potential locations for concentrators with known capacities, the goal is to install concentrators and assign each terminal to a concentrator so that the sum of the cost of installing concentrators and the cost of assigning terminals to concentrators is minimized (see, e.g., [8]). We identify vehicles with concentrators and demand points with terminals. If each terminal should be assigned to a single concentrator, the problem is called CCLP with single homing. This version is known to be equivalent to the capacitated facility location problem with single-source constraints (see, e.g., the survey paper [15] and the references therein). In contrast, since each demand point can be served by several vehicles, the MDPFAC is related to the CCLP with multiple homing.

There are very few studies related to the CCLP with multiple homing (see, e.g., [15,8,28]). Tang et al. [27] study the variant of the problem in which each terminal is assigned to a predetermined number of concentrators. The capacity of each concentrator is measured in terms of the number of terminals assigned to it. They first solve an assignment problem and propose a heuristic to account for multiple homing.

In another study, Pirkul et al. [23] consider the version of the CCLP where each terminal is assigned to two concentrators (primary and secondary). They propose a Lagrangian relaxation scheme to solve the problem. They extend their approach to the case where each terminal is assigned to k concentrators.

The MDPFAC differs from these studies in the following ways. First, there is no a priori restriction on the number of terminal-concentrator assignments. Rather, the optimization model (DA) determines the number of assignments so as to minimize the total assignment cost. Second, the description of our problem requires the introduction of logical constraints (10), which stipulate that a concentrator can be installed at location j if and only if terminal j is assigned to it. Finally, the multicommodity nature of the MDPFAC together with the indivisibility of each unit of each item further increases the difficulty of our problem.

Similarly, if each vehicle is viewed as a capacitated facility that can supply any combination of the items, the MDPFAC is related to the Multicommodity Capacitated Facility Location Problem (MCFLP). One of the earliest studies on MCFLP is due to Geoffrion and Graves [11]. Several variants of the problem have been studied in the literature (see, e.g., [14,17,12,20,5,21]). Typically, the problem is formulated as an instance of mixed integer linear programming and either a decomposition-based approach or a branch-and-bound scheme is employed together with Lagrangian relaxation. For a detailed account of the literature, we refer the reader to the books [7,8], the relatively recent survey papers [16,24,25,22], and the references therein.

Unlike a typical facility location problem, the variable transportation cost from a facility to a demand point in the MCFLP is replaced in the MDFAC by a fixed assignment cost. For instance, even if the facility locations are given in our problem, the resulting subproblem is a version of the many-to-many assignment problem (see, e.g., [19]), which is a further generalization of the NP-hard generalized assignment problem.

It follows that the MDPFAC can be viewed as a variant of the CCLP with multiple homing or of the multiproduct capacitated facility location problem. To the best of our knowledge, this particular variant of the network design problem has not been studied in the literature.

The optimization model (DA) consists of $mn+(m+2)\sum_{j=1}^{n} u_j + (m+1)\sum_{j=1}^{n}|\mathcal{O}_j|$ constraints, $m\sum_{j=1}^{n}|\mathcal{S}_j|u_j$ integer variables, and $\sum_{j=1}^{n}|\mathcal{O}_j|u_j + \sum_{j=1}^{n} u_j$ binary variables. Therefore, as illustrated by our computational results in Section 4, the size of (DA) quickly reaches beyond the capabilities of the current state-of-the-art solvers as the number of items m, the number of demand points n, and the number of potential vehicles $\sum_{j=1}^{n} u_j$ increase. In an attempt to circumvent this disadvantage of the optimization model (DA) especially for medium- to large-scale instances, we propose a hierarchical approach, which is the topic of the next section.

2.2. Hierarchical Approach

Our Hierarchical Approach consists of two stages. In the first stage, our main objective is to simplify the optimization model (DA) by reducing the number of item types and by a partial relaxation of the indivisibility assumption on each unit. In order to achieve this objective, we apply a specific demand aggregation scheme. To this end, we first compute the overall vehicle capacity requirement of each demand point, i.e., we define a new parameter

$$c_k := \sum_{i=1}^{m} w_i d_{ik}, \quad k=1,\ldots,n. \tag{15}$$

Next, we partition the total vehicle capacity requirement c_k of demand point k into two parts. The first part consists of the set of items that will be delivered to demand point k by a vehicle from class k, $k=1,\ldots,n$. Recall that each vehicle from class k is required to deliver at least one unit to demand point k. The remainder of the demand of demand point k to be served by vehicles from other classes constitutes the second part. In our aggregation scheme, we treat the first part of the demand exactly as in the direct formulation, i.e., no aggregation is used for the first part of the demand. On the other hand, for the second part of the demand, we no longer distinguish between different item types but rather treat this part as a single item type obtained by aggregating the demand for all different item types. We measure the demand corresponding to the second part simply in terms of the aggregate vehicle capacity requirements.

This partial aggregation scheme leads to a simplified optimization model, which is usually easier to solve than the direct formulation (DA). We adopt the same parameters defined in Section 2 and use the same binary variables y_{jkv} and z_{jv} in the simplified formulation. For the first part of the demand of demand point k, we still maintain the integral decision variables x_{ikkv}, $i=1,\ldots,m;\ k=1,\ldots,n;\ v=1,\ldots,u_k$. The integer variables x_{ijkv} corresponding to the second part of the demand of demand point k, however, are fully aggregated and replaced by the following *continuous* variables:

- s_{jkv}: the portion of the capacity of vehicle v from class j allocated to the second part of the demand of demand point k, $j=1,\ldots,n;\ k\in\mathcal{O}_j;\ v=1,\ldots,u_j$.

In addition, we define the following parameters:

$$\overline{w}_k := \min_{i=1,\ldots,m}\{w_i : d_{ik}>0\},\quad k=1,\ldots,n. \tag{16}$$

The parameter $\overline{w}_k$ is precisely the smallest capacity that should be reserved in any vehicle that serves demand point k. We obtain the following simplified mixed integer linear programming formulation:

$$\min\quad \sum_{j=1}^{n}\sum_{v=1}^{u_j} f_j z_{jv} + \sum_{j=1}^{n}\sum_{k\in\mathcal{O}_j}\sum_{v=1}^{u_j} o_{jk} y_{jkv} \tag{17}$$

$$\text{s.t.}\quad \sum_{i=1}^{m}\sum_{v=1}^{u_k} w_i x_{ikkv} + \sum_{j:k\in\mathcal{O}_j}\sum_{v=1}^{u_j} s_{jkv} = c_k,\ k=1,\ldots,n, \tag{18}$$

$$\sum_{v=1}^{u_k} x_{ikkv} \le d_{ik},\quad i=1,\ldots,m;\ k=1,\ldots,n, \tag{19}$$

$$\sum_{i=1}^{m} w_i x_{ijjv} + \sum_{k\in\mathcal{O}_j} s_{jkv} \le z_{jv},\quad j=1,\ldots,n;\ v=1,\ldots,u_j, \tag{20}$$

$$\text{(HA1)}\quad s_{jkv} \le y_{jkv},\quad j=1,\ldots,n;\ k\in\mathcal{O}_j;\ v=1,\ldots,u_j, \tag{21}$$

$$\sum_{i=1}^{m} x_{ijjv} \ge z_{jv},\quad j=1,\ldots,n;\ v=1,\ldots,u_j, \tag{22}$$

$$s_{jkv} \ge \overline{w}_k y_{jkv},\quad j=1,\ldots,n;\ k\in\mathcal{O}_j;\ v=1,\ldots,u_j, \tag{23}$$

$$s_{jkv} \ge 0,\quad j=1,\ldots,n;\ k\in\mathcal{O}_j;\ v=1,\ldots,u_j, \tag{24}$$

$$x_{ikkv} \ge 0,\quad i=1,\ldots,m;\ k=1,\ldots,n;\ v=1,\ldots,u_k, \tag{25}$$

$$x_{ikkv}\ \text{integer},\quad i=1,\ldots,m;\ k=1,\ldots,n;\ v=1,\ldots,u_k, \tag{26}$$

$$y_{jkv}\in\{0,1\},\quad j=1,\ldots,n;\ k\in\mathcal{O}_j;\ v=1,\ldots,u_j, \tag{27}$$

$$z_{jv}\in\{0,1\},\quad j=1,\ldots,n;\ v=1,\ldots,u_j. \tag{28}$$

Similar to (DA), the objective function (17) consists of the sum of individual fixed and overhead costs. The constraints (18) and (19) account for the first and second parts of the demand of each demand point. The counterparts of constraints (8)–(10) in (DA) are given by (20)–(22), respectively. The constraints (23) ensure that a minimum threshold capacity should be reserved for each demand point served by each vehicle from class j. Finally, the constraints (24) through (28) define the range of values for each variable.

The optimization model (HA1) consists of $n+mn+(m+2)\sum_{j=1}^{n} u_j+3\sum_{j=1}^{n}|\mathcal{O}_j|u_j$ constraints, $m\sum_{j=1}^{n} u_j$ integer variables, $\sum_{j=1}^{n}|\mathcal{O}_j|u_j+\sum_{j=1}^{n} u_j$ binary variables, and $\sum_{j=1}^{n}|\mathcal{O}_j|u_j$ continuous variables. Compared with (DA), the simplified model (HA1) consists of $(m-2)\sum_{j=1}^{n}|\mathcal{O}_j|u_j-n$ fewer constraints and $m\sum_{j=1}^{n}|\mathcal{O}_j|u_j$ fewer integer variables, which is a considerable reduction over the original formulation especially for larger instances. As such, (HA1) is considerably simpler to solve than (DA).

We remark that the optimization model (HA1) can be viewed as a partial relaxation of the original model (DA). Indeed, any feasible solution $(\tilde{x},\tilde{y},\tilde{z})$ of (DA) can be transformed into a feasible solution $(\overline{s},\overline{x},\overline{y},\overline{z})$ of (HA1) by defining

$$\overline{s}_{jkv} := \sum_{i=1}^{m} w_i \tilde{x}_{ijkv},\quad j=1,\ldots,n;\ k\in\mathcal{O}_j;\ v=1,\ldots,u_j,$$
$$\overline{x}_{ikkv} := \tilde{x}_{ikkv},\quad i=1,\ldots,m;\ k=1,\ldots,n;\ v=1,\ldots,u_k,$$
$$\overline{y}_{jkv} := \tilde{y}_{jkv},\quad j=1,\ldots,n;\ k\in\mathcal{O}_j;\ v=1,\ldots,u_j,$$
$$\overline{z}_{jv} := \tilde{z}_{jv},\quad j=1,\ldots,n;\ v=1,\ldots,u_j.$$

It follows from (7), (8), and the definitions (15) and (16) that $(\overline{s},\overline{x},\overline{y},\overline{z})$ is a feasible solution of (HA1) with the same objective function value as that of $(\tilde{x},\tilde{y},\tilde{z})$. We immediately obtain that

$$OPT(\text{HA1}) \le OPT(\text{DA}), \tag{29}$$

where $OPT(\cdot)$ denotes the optimal value of an optimization problem.

There are two differences between (HA1) and the optimization problem obtained from (DA) by relaxing the integer variables x_{ijkv}, $i=1,\ldots,m;\ j=1,\ldots,n;\ k\in\mathcal{O}_j;\ v=1,\ldots,u_j$. First, the use of aggregated continuous variables s_{jkv} significantly decreases the number of variables, which may contribute to the solvability of the model. Second, the additional constraints (23) may lead to a stronger relaxation.

The solution of the first stage determines the number of vehicles to be used from each class j, $j=1,\ldots,n$, the set of items to be delivered to demand point j by each vehicle from class j, the set of demand points $k\in\mathcal{O}_j$ to be served by each vehicle from class j, and the portion of the capacity allocated to each demand point $k\in\mathcal{O}_j$ in each vehicle from class j. Note that, for each demand point j, the first part of the demand is explicitly allocated in each vehicle from class j whereas only the aggregate capacity requirement in each vehicle from every other class is returned for the second part of the demand. An important observation is that the way in which the demand of each demand point is divided between the two parts is entirely determined by an optimal solution of (HA1).

The aggregation of the second part of the demand of each demand point may lead to an optimal solution in which the number of vehicles and allocated capacities in each vehicle may not exactly accommodate the (second part) of the demand of a demand point for individual item types due to the relaxation of the indivisibility assumption (see Example 3.1 in Section 3). In an attempt to overcome this problem, we design the second stage aiming to disaggregate the second part of the demand of each demand point into individual item types while trying to somehow respect the solution of the first stage. Recall that the solution of

the first stage determines the set of vehicles to be used, the set of demand points to be served by each vehicle, and the capacity of each vehicle allocated to the second part of the demand of each demand point. In the second stage, we aim to compute a solution that respects the first two of these decisions while completely ignoring the third one. Such an approach enables us to potentially circumvent the aforementioned infeasibility problem that may arise from the conflict between the reintroduction of the integrality constraints for each individual item type and the aggregately computed capacity allocations in each vehicle in the first stage.

Let us define the following additional parameters:

- x^*_{ikkv}: the optimal value of x_{ikkv} in (HA1), $i=1,\ldots,m$; $k=1,\ldots,n$; $v=1,\ldots,u_k$;
- y^*_{jkv}: the optimal value of y_{jkv} in (HA1), $j=1,\ldots,n$; $k\in\mathcal{O}_j$; $v=1,\ldots,u_j$;
- z^*_{jv}: the optimal value of z_{jv} in (HA1), $j=1,\ldots,n$; $v=1,\ldots,u_j$;
- $\mathcal{V}^*_j$: the set of vehicles from class j to be used in transportation, $j=1,\ldots,n$, i.e.,
$$\mathcal{V}^*_j=\{v\in\{1,2,\ldots,u_j\}: z^*_{jv}=1\},\quad j=1,\ldots,n;$$
- $\mathcal{O}^*_{jv}$: the set of demand points different from j served by vehicle $v\in\mathcal{V}^*_j$, $j=1,\ldots,n$; $v\in\mathcal{V}^*_j$, i.e.,
$$\mathcal{O}^*_{jv}=\{k\in\mathcal{O}_j: y^*_{jkv}=1\},\quad j=1,\ldots,n;\ v\in\mathcal{V}^*_j.$$

Observe that each of these parameters is obtained using the solution of (HA1) in the first stage. We now introduce the decision variables for the optimization problem to be solved in the second stage.

- x_{ijkv}: the number of units of item type i delivered to demand point k by vehicle $v\in\mathcal{V}^*_j$, $i=1,\ldots,m$; $j=1,\ldots,n$; $v\in\mathcal{V}^*_j$; $k\in\mathcal{O}^*_{jv}$;
- a: the largest excess capacity required on any vehicle.

The optimization problem is presented below:

$$\min\quad a \tag{30}$$

$$\sum_{j:k\in\mathcal{O}_j}\ \sum_{v\in\mathcal{V}^*_j:k\in\mathcal{O}^*_{jv}} x_{ijkv}=d_{ik}-\sum_{v\in\mathcal{V}^*_k} x^*_{ikkv},\quad i=1,\ldots,m;\ k=1,\ldots,n, \tag{31}$$

$$\text{(HA2)}\quad \sum_{i=1}^{m}\sum_{k\in\mathcal{O}^*_{jv}} w_i x_{ijkv}\le\left(1-\sum_{i=1}^{m} w_i x^*_{ijjv}\right)+a,\quad j=1,\ldots,n;\ v\in\mathcal{V}^*_j, \tag{32}$$

$$a\ge 0, \tag{33}$$

$$x_{ijkv}\ge 0,\quad i=1,\ldots,m;\ j=1,\ldots,n;\quad v\in\mathcal{V}^*_j;\ k\in\mathcal{O}^*_{jv}, \tag{34}$$

$$x_{ijkv}\ \text{integer},\quad i=1,\ldots,m;\ j=1,\ldots,n;\quad v\in\mathcal{V}^*_j;\ k\in\mathcal{O}^*_{jv}. \tag{35}$$

The constraint set (30) ensures that the demand of each demand point for each item type is exactly satisfied. The capacity restriction for each vehicle is formulated by (32). Observe that this constraint set allows us to exceed the capacity of each vehicle by the value of the nonnegative decision variable a, which, however, is minimized by the objective function (31). The constraints (33) through (34) define the range of values each decision variable can take.

The optimization model (HA2) is used to determine the allocation of the second part of the demand of each demand point at the level of individual item types. Despite the fact that capacity allocations in vehicles from the first stage are not explicitly taken into account in the second stage, the optimal value of (HA2) can still be strictly positive. In this case, there exists no feasible disaggregation of the second part of the demand of demand points that exactly respects the set of vehicles and the demand point–vehicle assignments returned by the first stage. We discuss this issue in more detail in Section 3. Nevertheless, the optimization model (HA2) aims to compute a solution that closely resembles the solution of (HA1) while putting an emphasis on feasibility for the original problem.

Before closing this section, we make an important observation about (HA2). Based on the solution of (HA1), the second stage problem aims to allocate the second part of the demand of each demand point for different item types among vehicles that have a positive capacity allocation for that particular demand point. Therefore, we can construct a graph $G=(V,E)$ such that V consists of the demand points k, $k=1,\ldots,n$, and vehicles $v\in\mathcal{V}^*_j$, $j=1,\ldots,n$. There is an edge between demand point k and vehicle $v\in\mathcal{V}^*_j$ if $k\in\mathcal{O}^*_{jv}$, i.e., if vehicle $v\in\mathcal{V}^*_j$ has a positive capacity allocation for the second part of the demand of demand point k. By construction, G is a bipartite graph. If G is not connected, then the problem (HA2) can be solved independently for each connected component since (HA2) then naturally decomposes into smaller optimization problems. As illustrated by our computational experiments detailed in Section 4, this decomposition may lead to considerable savings in the solution of (HA2). For simplicity, we will continue to refer to the optimization model (HA2) as a single optimization problem with the implicit understanding that it can be decomposed into smaller problems.

3. Analysis of the Hierarchical Approach

In this section, we analyze the solution computed by the Hierarchical Approach. We develop conditions under which such a solution is an optimal solution of the MDPFAC. In addition, we establish an upper bound on an infeasibility measure of this solution with respect to the original problem. Finally, if the solution computed by the Hierarchical Approach happens to be infeasible for the MDPFAC, we propose a scheme that computes a feasible solution of good quality.

We first provide a sufficient condition under which the solution returned by the Hierarchical Approach is actually an optimal solution of the original problem.

Lemma 3.1. *Let* (s^*,x^1,y^*,z^*) *be any optimal solution of* (*HA1*). *Suppose that* $OPT(HA2)=0$ *and let* (a^*,x^2) *denote any optimal solution of* (*HA2*) *corresponding to the optimal solution* (s^*,x^1,y^*,z^*) *of* (*HA1*). *Then*, (x^*,y^*,z^*) *is an optimal solution of* (*DA*), *with the convention that*

$$x^*_{ikkv}=x^1_{ikkv},\quad i=1,\ldots,m;\ k=1,\ldots,n;\ v=1,\ldots,u_k,$$
$$x^*_{ijkv}=x^2_{ijkv},\quad i=1,\ldots,m;\ j=1,\ldots,n;\ v\in\mathcal{V}^*_j;\ k\in\mathcal{O}^*_{jv},$$

and $x^*_{ijkv}=0$ *for each* (i,j,k,v) *that appears in* (*DA*) *but not in* (*HA2*).

Proof. Under the hypotheses of the lemma, it follows from the construction of (HA1) and (HA2) that the solution (x^*,y^*,z^*) is feasible for (DA). Since (s^*,x^1,y^*,z^*) is an optimal solution of (HA1) and since (DA) and (HA1) have the same objective function, it follows that $OPT(\text{DA})\le OPT(\text{HA1})$. The assertion is established by combining this inequality with (29). □

Under the hypotheses of Lemma 3.1, there exists a feasible solution of the original problem that respects the subset of vehicles used in transportation and the vehicle–demand point

assignments in the solution obtained by our aggregation scheme. However, as illustrated by the following Example 3.1, such a feasible solution might not exist in general.

Example 3.1. Suppose that there is a single item type with $w_1 = 2/3$ and there are two demand points 1 and 2. Assume that $d_{11} = 2$ and $d_{12} = 1$, i.e, demand point 1 places an order for 2 units whereas demand point 2 orders one unit of this item. Let $f_1 = 2$, $f_2 = 1$, and $o_{12} = 0.1$. Note that $\mathcal{O}_1 = \{2\}$ and $\mathcal{O}_2 = \emptyset$. It is easy to see that the optimal solution of (HA1) uses two vehicles from class 1 and each one delivers one full unit to demand point 1. The single item ordered by demand point 2 is equally split between the two vehicles. Therefore, $OPT(\text{HA1}) = 2(2) + 2(0.1) = 4.2$. However, in the second stage, the best solution that can be computed using only these two vehicles from class 1, each of which is loaded with one unit, is given by loading the item ordered by demand point 2 to either one of the vehicles. The resulting solution is clearly infeasible for the original problem since we need to exceed the capacity of one of the vehicles by 1/3. In this case, $OPT(\text{HA2}) = 1/3$. On the other hand, the optimal solution of the original problem uses three separate vehicles, two from class 1 and one from class 2, each of which delivers exactly one unit of the item to the respective demand points, i.e., $OPT(\text{DA}) = 5$.

Example 3.1 illustrates a simple instance with $OPT(\text{HA1}) < OPT(\text{DA})$, which implies that the solution returned by the Hierarchical Approach is infeasible for the original problem, resulting in $OPT(\text{HA2}) > 0$. We next establish an upper bound on $OPT(\text{HA2})$. This bound provides an explicit characterization of the quality of such a solution in terms of infeasibility with respect to the vehicle capacity constraints in the original problem.

We first establish an interesting connection between the Hierarchical Approach and the makespan scheduling problem on unrelated parallel machines. Given a finite number of jobs $i \in \mathcal{I}$ and a finite number of parallel machines $v \in \mathcal{V}$, where the processing time of job i on machine v is given by $p_{iv}, i \in \mathcal{I}, v \in \mathcal{V}$, the minimum makespan scheduling problem asks for an assignment of each job to exactly one machine in such a way that the maximum total processing time on any machine, called the makespan, is as small as possible.

In the context of the MDPFAC, we can view each unit of each item as a separate job and each vehicle as a machine. Any optimal solution of (HA1) determines the set of machines that can be used to process each job as well as the processing time of each job on each machine. More precisely, let $\mathcal{I}_k$ denote the set of items ordered by demand point k, $k = 1, \ldots, n$. Recall that an optimal solution of (HA1) induces a partition of $\mathcal{I}_k$ into two subsets. Let $\mathcal{I}_k^{(1)}$ and $\mathcal{I}_k^{(2)}$ denote the set of items designated to be in the first and second part of the demand, respectively, so that $\mathcal{I}_k = \mathcal{I}_k^{(1)} \cup \mathcal{I}_k^{(2)}$, $k = 1, \ldots, n$. Note that each item in $\mathcal{I}_k^{(1)}$ is assigned to exactly one vehicle $v \in \mathcal{V}_k^*$. On the other hand, an item in $\mathcal{I}_k^{(2)}$ can be assigned to any vehicle $v \in \mathcal{V}_j^*$, $j = 1, \ldots, n$ as long as $k \in \mathcal{O}_{jv}^*$, where $\mathcal{V}_j^*$ and $\mathcal{O}_{jv}^*$ are as defined in Section 2.2. Therefore, the optimal solution of (HA1) induces the following processing times for each job: for a given item $i \in \mathcal{I}_k^{(1)}$, the processing time p_{iv} on machine $v \in \mathcal{V}_k^*$ is w_i if item i is assigned to vehicle $v \in \mathcal{V}_k^*$ in the optimal solution of (HA1) and $p_{iv} = +\infty$ for each of the remaining machines. Similarly, for an item $i \in \mathcal{I}_k^{(2)}$ and a vehicle $v \in \mathcal{V}_j^*$, we define the processing time $p_{iv} = w_i$ if $k \in \mathcal{O}_{jv}^*$ and $p_{iv} = +\infty$ for each of the remaining machines. Using this reformulation, the relation between the second stage problem (HA2) and this particular instance of the makespan scheduling problem becomes evident: it follows that $OPT(\text{HA2}) = 0$ if and only if there exists a feasible schedule whose makespan is less than or equal to one. On the other hand, a positive optimal value of (HA2) implies that the minimum makespan is given by $1 + OPT(\text{HA2})$. This is the key observation that allows us to obtain the subsequent results.

The next theorem establishes an upper bound on the optimal value of the optimization problem (HA2), which corresponds to the largest "excess" capacity required on any vehicle in order to accommodate the demand of all demand points.

Theorem 3.1. *We have that*

$$OPT(\text{HA2}) \le \left(1 - \frac{1}{|\mathcal{V}^*|}\right) \max_{i=1,\ldots,m} w_i, \tag{36}$$

where

$$\mathcal{V}^* := \bigcup_{j=1}^{n} \mathcal{V}_j^*.$$

Proof. Note that $|\mathcal{V}^*|$ is the total number of vehicles that can be used in the second stage. If $|\mathcal{V}^*| = 1$, then the first stage model (HA1) returns a solution in which the combined demand of all the demand points can be loaded on a single vehicle. Since we assume that any combination of the item types can be loaded on a single vehicle as long as the total required vehicle capacity does not exceed the vehicle capacity, which is assumed to be one, it follows that any optimal solution of (HA2) is a feasible solution of the original problem. Therefore, $OPT(\text{HA2}) = 0$, which satisfies (36). Let us therefore assume that $|\mathcal{V}^*| \ge 2$.

We establish the upper bound (36) by constructing an appropriate feasible solution of (HA2). Let us first consider the following linear programming problem:

$$\begin{aligned}
(\text{LP}) \quad \min \quad & d \\
\text{s.t.} \quad & \sum_{i \in \mathcal{I}} p_{iv} x_{iv} \le d, \quad v \in \mathcal{V}, \\
& \sum_{v \in \mathcal{V}} x_{iv} = 1, \quad i \in \mathcal{I}, \\
& x_{iv} \ge 0, \quad i \in \mathcal{I},\ v \in \mathcal{V},
\end{aligned}$$

where $\mathcal{I}$ and $\mathcal{V}$ are given finite sets, d and $x_{iv}, i \in \mathcal{I}, v \in \mathcal{V}$ are the decision variables, and $p_{iv}, i \in \mathcal{I}, v \in \mathcal{V}$ are the nonnegative parameters. Note that (LP) can be viewed as a relaxation of the makespan scheduling problem on unrelated parallel machines with appropriate definitions of the parameters.

A b-balanced (fractional) schedule is a feasible solution of (LP) such that $d = b$ and $p_{iv} \le b$ for each $x_{iv} > 0$ (see [26]). Therefore, a b-balanced (fractional) schedule allows for the (partial) assignment of jobs to machines in such a way that no job $i \in \mathcal{I}$ is partially or fully assigned to a machine $v \in \mathcal{V}$ if its processing time p_{iv} satisfies $p_{iv} > b$ and the makespan of this (fractional) schedule is at most b. Let b^* denote the smallest value of b such that a b-balanced (fractional) schedule exists. Note that the optimal value of (LP) is a lower bound on b^*.

We next show that an optimal solution of (HA1) can be turned into a feasible solution $\bar{x}$ of an instance of (LP). Consider an optimal solution of (HA1). For each item $i \in \mathcal{I}_k^{(1)}$, define $\bar{x}_{iv} = 1$ if that item is assigned to vehicle $v \in \mathcal{V}_k^*$. Next, distribute each item $i \in \mathcal{I}_k^{(2)}$ in a greedy manner to vehicles $v \in \mathcal{V}_j^*$ such that $k \in \mathcal{O}_{jv}^*$, without exceeding the allocated capacity for that demand point on each vehicle. Set $\bar{x}_{iv} = 0$ for all the remaining (i,v) pairs. Since the total allocated capacity equals the total demand and since the items in $\mathcal{I}_k^{(2)}$ are allowed to be split in any fraction, this procedure yields a feasible solution to (LP), where $\mathcal{I} = \bigcup_{k=1}^{n} \mathcal{I}_k$ is the set of all items to be delivered to demand points, $\mathcal{V} = \mathcal{V}^* = \bigcup_{j=1}^{n} \mathcal{V}_j^*$ is the set of all vehicles returned by the solution of (HA1), and p_{iv} is defined as in the discussion preceding the theorem. For a given item $i \in \mathcal{I}$ and machine $v \in \mathcal{V}$, we can have $\bar{x}_{iv} > 0$ in the resulting feasible solution of (LP) if and only if $p_{iv} = w_i$. Since $\max_{i=1,\ldots,m} w_i \le 1$, it follows that

this particular feasible solution is a b-balanced (fractional) schedule for the corresponding makespan scheduling problem with $b=1$, which implies that $b^* \leq 1$ on this instance. Shchepin and Vakhania [26] propose a procedure that first computes a particular feasible solution corresponding to a b^*-balanced (fractional) schedule. Using a sophisticated procedure, this feasible solution is then "rounded" to produce a feasible (integral) schedule for the makespan scheduling problem whose makespan is at most $b^*+(1-1/|\mathcal{V}^*|)(\max_{i=1,\dots,m} w_i)$. Since $b^* \leq 1$, it follows that such a schedule corresponds to a feasible solution of (HA2) with $a \leq (1-1/|\mathcal{V}^*|)(\max_{i=1,\dots,m} w_i)$. The assertion follows. □

Theorem 3.2 provides a characterization of the quality of the solution returned by the Hierarchical Approach, where the quality is measured in terms of infeasibility with respect to vehicle capacity constraints. For instance, if the relative item sizes w_i are small, then the solution of the Hierarchical Approach would necessarily have a small infeasibility measure.

We remark that the proof of Theorem 3.1 heavily relies on the aforementioned connection between our problem and the makespan scheduling problem. It turns out that this bound, in general, cannot be further improved. Consider the instance in Example 3.1. We have $|\mathcal{V}^*|=2$ and $\max_{i=1,\dots,m} w_i = 2/3$. Therefore, the upper bound of Theorem 3.1 is given by $(1-1/2)(2/3)=1/3$, which matches the optimal value $OPT(\text{HA2})=1/3$. It follows that this upper bound is tight.

Note that Lemma 3.1 gives a sufficient condition under which the solution returned by the Hierarchical Approach is actually optimal for the original problem. Theorem 3.1 establishes a tight upper bound on a measure of infeasibility of the solution computed by the Hierarchical Approach for the original problem. We close this section by the following result, which essentially proposes an algorithm that constructs a "good" feasible solution of the original problem if the solution computed by the Hierarchical Approach happens to be infeasible (i.e., if $OPT(\text{HA2})>0$).

Theorem 3.2. *Suppose that $OPT(HA2)>0$. Using an optimal solution of (HA1), one can construct a feasible solution of the original problem whose objective function value is at most $2OPT(DA)$.*

Proof. The proof relies on the following "rounding" procedure due to Lenstra et al. [18]. Consider the following system of finite linear equalities and inequalities:

$$\sum_{v \in \mathcal{V}_i} x_{iv} = 1, \quad i \in \mathcal{I},$$
$$\sum_{i \in \mathcal{I}_v} p_{iv} x_{iv} \leq 1, \quad v \in \mathcal{V},$$
$$x_{iv} \geq 0, \quad v \in \mathcal{V},\ i \in \mathcal{I}_v, \tag{37}$$

where $\mathcal{I}$ and $\mathcal{V}$ are given finite sets, $x_{iv}, i \in \mathcal{I}, v \in \mathcal{V}$ are the decision variables, p_{iv}, $i \in \mathcal{I}$, $v \in \mathcal{V}$ are nonnegative parameters, and

$$\mathcal{V}_i := \{v \in \mathcal{V} : p_{iv} \leq 1\}, \quad i \in \mathcal{I};\ \mathcal{I}_v := \{i \in \mathcal{I} : p_{iv} \leq 1\},\ v \in \mathcal{V}. \tag{38}$$

If the system (37) is feasible, one can compute a vertex of this polytope (in time which is polynomial in the size of the input, see, e.g., [13]).

Similar to the proof of Theorem 3.1, any optimal solution of (HA1) can be transformed into a feasible solution of the system (37), where the parameters p_{iv} are defined in the same way and the sets $\mathcal{I}$ and $\mathcal{V}$ are identified with jobs and machines, respectively. Let $\tilde{x}$ denote a vertex of the corresponding polytope. Let us consider $\tilde{x}_{iv}$ as the value of the (partial) assignment of job $i \in \mathcal{I}$ to machine $v \in \mathcal{V}$. Note that $\tilde{x}$ has $|\mathcal{I}|+|\mathcal{V}|$ basic variables. Since at least one $\tilde{x}_{iv}$ should be basic for each job $i \in \mathcal{I}$, it follows that at most $|\mathcal{V}|$ jobs have fractional values and are therefore split into more than one machine by the solution $\tilde{x}$. In [18], a bipartite graph is constructed whose vertices are given by the sets $\mathcal{I}$ and $\mathcal{V}$. There is an edge between $i \in \mathcal{I}$ and $v \in \mathcal{V}$ if $\tilde{x}_{iv} > 0$. Next, a matching that covers each node corresponding to a split job is constructed on the subset of the edges (i,v) for which $0 < \tilde{x}_{iv} < 1$. The rounding procedure assigns split jobs to machines using this matching. The matching procedure ensures that each partially assigned job is fully assigned to a single machine and each machine receives at most one of the split jobs. Therefore, in the rounded schedule, there are no split jobs and each machine receives at most one full job, which was partially assigned to it by $\tilde{x}$. The completion time on each machine increases by at most the processing time of this rounded job. It follows that, in the resulting assignment, the excess capacity in each vehicle can only be caused by a single item that was initially split with respect to $\tilde{x}$ and was later rounded to a full item on this vehicle.

We finally transform the resulting solution into a feasible solution satisfying vehicle capacity constraints. Let us focus on a vehicle v^* whose capacity is exceeded by this procedure and let us denote the single item that leads to the capacity violation by $i^* \in \mathcal{I}_{k^*}$. We now remove this single item from the vehicle v^* and move it to a new vehicle from class k^*, which increases the total cost by f_{k^*}. Note that v^* either belongs to class k^* or to a class j such that $k^* \in \mathcal{O}^*_{jv^*}$. In the latter case, it follows from (1) and the definition of $\mathcal{O}^*_{jv^*}$ that the fixed cost f_{k^*} of the new vehicle is bounded above by the fixed cost f_j incurred by v^*. In the first case, if v^* still belongs to class k^* after the removal of item i^*, then the fixed cost of the new vehicle is bounded above by the fixed cost incurred by v^*. If, on the other hand, i^* was the only item on v^* that was ordered by demand point k^*, then v^* now belongs to a different class $k^{**} \in \mathcal{O}_{k^*}$. In the new solution, the total fixed cost is $f_{k^{**}}+f_{k^*} \leq 2f_{k^*}$ since $f_{k^{**}} \leq f_{k^*}$, and the overhead cost of v^* decreases by $o_{k^*,k^{**}}$. Therefore, in all cases, the increase in the total cost is bounded above by the fixed cost of the original vehicle v^*. Furthermore, this procedure ensures that the capacity constraints on both v^* and the new vehicle are now satisfied. We repeat this procedure for each vehicle whose capacity is violated by the rounding procedure. It follows that the resulting solution is feasible for the original problem with a total cost bounded above by $2OPT(\text{HA1})$. Together with (29), we have $2OPT(\text{HA1}) \leq 2OPT(\text{DA})$, which concludes the proof. □

The proof of Theorem 3.2 is constructive and yields a feasibility restoration procedure using an optimal solution of (HA1) if the sufficient condition of Lemma 3.1 does not hold. We do not know if the approximation factor 2 is tight in the analysis. However, the following simple example illustrates that the procedure can asymptotically double the number of vehicles returned by (HA1) and yield a suboptimal solution.

Example 3.2. Suppose that there are two demand points 1 and 2 and there are two item types, a "large" item and a "small" item with $w_1 = 1/2+1/(2p)$ and $w_2 = 1/2$, respectively, where p is a large positive integer. Let $f_1 = f_2 = 1$ and $o_{12} = 1/(2p)$. We have $\mathcal{O}_1 = \{2\}$ and $\mathcal{O}_2 = \emptyset$. Suppose that demand point 1 orders $2p$ units of the large item and demand point 2 orders $2(p-1)$ units of the small item. The overall vehicle capacity requirement of both demand points is $2p$. It is easy to verify that there is an optimal solution of (HA1) that uses $2p$ vehicles from class 1 such that each unit of the large item is loaded onto each of the $2p$ vehicles. The $2(p-1)$ small items ordered by demand point 2 are split using the residual capacities in each of the vehicles. It follows that

$OPT(HA1) = 2p(1) + 2p(1/(2p)) = 2p + 1$. Note that each small item is split between at least two vehicles since the residual capacity in each vehicle is smaller than the size of the smaller item. Our procedure assigns each of the split $2p-2$ items to a separate vehicle, which results in $2p-2$ vehicles each of whose capacity is violated. For each such vehicle, a new vehicle is introduced. Therefore, our procedure uses a total of $2p$ vehicles from class 1 and $2p-2$ vehicles from class 2, yielding a total of $4p-2$ vehicles, which is asymptotically twice the number of vehicles used by (HA1). The cost of the solution returned by the feasibility restoration procedure is $4p-2$. The optimal solution of the original problem is given by loading each large item ordered by demand point 1 to a vehicle from class 1 and each pair of small items ordered by demand point 2 to a vehicle from class 2, thereby using a total of $2p+(p-1) = 3p-1$ vehicles, which implies that $OPT(DA) = 3p-1$. This example illustrates that the approximation factor of our procedure is at least 4/3.

We remark that the feasibility restoration procedure outlined in the proof of Theorem 3.2 does not necessarily yield a polynomial-time two-approximation algorithm since it still requires the computation of an optimal solution of the mixed integer linear programming problem (HA1). However, as illustrated in the next section, (HA1) is usually significantly easier to solve than (DA).

4. Computational results

In this section, we report our computational results using the direct formulation approach (henceforth the Direct Approach) and the Hierarchical Approach on a data set adapted from a major automotive manufacturer in Turkey.

Let us describe the data set in detail. The automotive manufacturer has a central depot which houses a large number of different spare parts and there are 63 geographically dispersed retailers (demand points), each of which places orders for spare parts on a daily basis. Upon receiving the order, the demand of each retailer is packed into appropriate boxes. The manufacturer uses a total of 71 different types of boxes for the shipment of spare parts. We therefore consider the demand of each retailer in terms of different types of boxes. We treat each box type as a different item type and assume that each box is an indivisible unit.

The physical distribution operations are carried out by a separate transportation company. However, the manufacturer is fully in charge of deciding how to serve the demand of each demand point. The transportation company charges the manufacturer based on the resulting solution according to the cost structure detailed in Section 2. The transportation company is responsible for providing any number of vehicles requested by the manufacturer. We use the procedure described in Section 2.1 in order to define a sufficiently large number of potential vehicles for our optimization models.

In our experiments, we used the algebraic modeling language GAMS 23.6.5, which uses CPLEX 12.2.0.2 as the mixed integer programming solver. The input data is imported from an electronic spreadsheet using a code written in Visual Basic. The code sets up the optimization problems, solves them and outputs the results into an electronic spreadsheet in a user-friendly format. The computational experiments were carried out on a notebook computer with 3 GB RAM and Intel Core 2 Duo 2.53 GHz P8700 processor running under 32-bit Windows 7.

For each optimization problem, we set a time limit of 6 h, which seems to be reasonable given that the MDPFAC needs to be solved on a daily basis. If the optimization problem cannot be solved within this time limit, the best feasible solution found by the solver is reported. Furthermore, the settings in CPLEX were configured in such a way that memory problems are circumvented to the largest extent possible. More specifically, we ensured that the node files were stored on the hard disk as opposed to the memory by setting the parameter `nodefileind` to 3 (node file on disk and compressed). In addition, we set both parameters `solvefinal` and `names` to 0, which results in storing no dual information and skipping the names of the constraints and variables during the solve, respectively. Apart from these modifications, the default settings have been adopted. We used the same settings for all optimization problems.

Our preliminary experiments revealed that the inclusion of the valid inequalities

$$y_{jk\nu} \le z_{j\nu}, \quad j = 1,\ldots,n;\ k \in \mathcal{O}_j;\ \nu = 1,\ldots,u_j \tag{39}$$

considerably improved the solution time. Therefore, we included all of these inequalities in both optimization models (DA) and (HA1).

For our experiments, we obtained the detailed data set for 11 consecutive days. Table 1 presents, for each of the 11 days, the number of demand points that placed an order (n), the total number of different item types ordered (m), average number of different item types ordered by each demand point, denoted by t, and the total number of vehicles ($|\mathcal{V}|$) resulting from the procedure in Section 2.1 on that particular day.

As illustrated by Table 1, there is a significant variation in terms of the number of demand points placing an order (n) and the average number of different item types ordered by each demand point (t), which has a direct influence on the size of the resulting optimization problems (DA), (HA1), and (HA2) as presented in Table 2. Note that these sizes correspond to the "Reduced MIP Statistics" reported by the solver CPLEX, i.e., the sizes of the problems sent to the solver after the presolve step.

Table 2 reveals that the size of the optimization model (HA1) is usually significantly smaller than that of (DA). Note that the reduction in the number of columns, which corresponds to the number of variables, is much more pronounced than the reduction in the number of rows, which corresponds to the number of constraints. This is an expected result since the aggregation scheme that leads to (HA1) significantly reduces the number of variables that corresponds to different item types. In addition, while each decision variable of (DA) is binary or integer-valued, the optimization model (HA1) contains a mixture of binary, integer-valued, and continuous variables, which is a further advantage of the Hierarchical Approach over the Direct Approach. We remark that the optimization model (DA) could not be sent to the solver on Day 11 due to memory issues despite our aforementioned precautions.

We also remark that the size of the optimization problem (HA2) is considerably smaller than each of the optimization models (DA) and (HA1). Furthermore, as explained at the end of Section 2.2, (HA2) can be decomposed into a number of further smaller optimization problems, which leads to even faster

Table 1
Summary of the input data.

Day	n	m	t	$\lvert\mathcal{V}\rvert$
1	14	45	14.64	77
2	11	50	16.82	39
3	12	47	16.92	46
4	17	49	16.35	104
5	10	37	11.70	24
6	19	44	14.68	63
7	23	49	16.43	163
8	10	49	18.10	40
9	7	39	17.86	26
10	8	41	20.25	34
11	27	48	16.56	175

Table 2
The sizes of the optimization models (DA), (HA1), and (HA2).

Day	# of rows			# of columns			# of nonzeros		
	(DA	(HA1)	(HA2)	(DA)	(HA1)	(HA2)	(DA)	(HA1)	(HA2)
1	4411	3376	0	37 884	3347	0	93 187	13 234	0
2	1847	1416	0	16 965	1416	0	41 842	5566	0
3	2334	1806	31	20 516	1869	85	51 086	7376	171
4	7200	5495	4	68 952	5141	22	169 265	20 356	44
5	1049	807	0	6936	804	0	17 172	3156	0
6	4996	3776	2	41 454	3325	11	102 510	13 145	22
7	15 166	11 485	0	140 669	10 005	0	348 246	39 694	0
8	1718	1333	0	15 200	1556	0	38 212	6133	0
9	810	639	0	6240	827	0	16 053	3245	0
10	1196	952	0	10 064	1214	0	26 210	4788	0
11	!	14 458	0	!	12 297	0	!	48 828	0

!: Problem instance could not be sent to the solver because of memory limitations.

Table 3
Second stage statistics.

Day	1	2	3	4	5	6	7	8	9	10	11
# of subproblems	5	2	4	6	3	6	12	2	1	4	10
# of subproblems solved in the presolve	5	2	3	4	3	5	12	2	1	4	10

Table 4
Comparison of direct approach and Hierarchical Approach.

Day	Direct Approach			Hierarchical Approach				
	Total cost	Gap (%)	CPU time (s)	Total cost	Gap (%)	CPU time (s)		
						(HA1)	(HA2)	Total
1	8236.20	0.00	103.74	8236.20	0.00	53.12	0.14	53.26
2	8221.71	0.00	23.60	8221.71	0.00	2.25	0.05	2.30
3	10 308.24	0.00	27.81	10 308.24	0.00	9.67	0.14	9.81
4	11 141.19	0.00	3240.42	11 141.19	0.00	413.93	0.15	414.08
5	4276.35	0.00	5.96	4276.35	0.00	2.31	0.08	2.39
6	10 141.40	0.00	647.15	10 141.40	0.00	20.90	0.14	21.04
7	13 550.70	13.99	132.29 (!)	12 887.65	0.00	1887.03	0.39	1887.42
8	7704.08	0.00	81.39	7704.08	0.00	8.13	0.06	8.19
9	3353.59	0.00	43.45	3353.59	0.00	7.55	0.02	7.57
10	3956.86	0.00	50.51	3956.86	0.00	12.78	0.08	12.85
11	!	!	12.04 (!)	13 613.84	0.00	15 828.96	0.29	15 829.25

(!) The solver stopped with "out of memory" status.

solution of the second stage of the Hierarchical Approach. For (HA2), Table 2 presents the total number of rows, columns, and nonzeros obtained from the sum of corresponding statistics for each smaller problem resulting from the decomposition. In particular, for the eight instances with 0 rows and 0 columns under the heading (HA2), each of the smaller problems was already solved at the presolve stage.

Our computational experiments revealed that the decomposition of (HA2) paid off also on the remaining three days. Table 3 presents, for each day, the number of subproblems resulting from the decomposition of (HA2) and the number of subproblems solved during the presolve stage.

We compare the Direct Approach with the Hierarchical Approach on the basis of the running time and the quality of the solution. Recall that the Direct Approach requires the solution of the single optimization problem (DA) whereas the Hierarchical Approach requires the solution of (HA1), followed by (HA2).

The solution times are the resource usage times returned by GAMS. For the Direct Approach, this is simply the time it takes to solve (DA) while we report the sum of the solution times of (HA1) and (HA2) for the Hierarchical Approach.

In Table 4, the results of the two solution approaches are reported in terms of solution times and the best solutions returned within the time limit. "Gap" corresponds to the relative gap returned by CPLEX, given by $(BestUB-BestLB)/|BestUB|$, where *BestUB* and *BestLB* denote the best upper and lower bounds on the optimal value, respectively. In terms of the CPU time, Table 4 reveals that the overall CPU time required by the Hierarchical Approach is consistently smaller than that of the Direct Approach. In some cases, the Hierarchical Approach is about 10 times faster than the Direct Approach. Such a drastic improvement is most likely due to the elimination of a significant number of integer variables, which also leads to a significant reduction in the total number of variables. In addition, both models (HA1) and (HA2) are solved to optimality within the time limit on each of the 11 days whereas the Direct Approach is terminated due to memory limitations on Days 7 and 11 despite our precautions regarding the memory use. It is worth noting that even the linear

programming relaxation of (DA) could not be solved on Day 11 due to the size of the problem. Another interesting observation is that the solution time of the second stage model (HA2) is less than half a second on each of the 11 days. This is a consequence of our previous observation that most of the subproblems of the second stage can already be solved in the presolve step of the solver (see Table 3). Given that this distribution problem needs to be solved on a daily basis, these results clearly indicate the advantage of the Hierarchical Approach over the Direct Approach in terms of the CPU time.

We next compare the two approaches in terms of the best solutions returned within the time limit. For the Hierarchical Approach, an important observation is that the optimal objective function value of (HA2) is found to be zero for each of the 11 days. Therefore, it follows from Lemma 3.1 that the solution computed by the Hierarchical Approach is actually optimal for the original problem on each of the 11 days. This is another clear advantage of the Hierarchical Approach on this data set.

Our computational results illustrate that both approaches yield the same optimal total cost on each day except for Days 7 and 11, on which the Direct Approach is terminated due to memory issues. On Day 7, the total cost of the best solution returned by the Direct Approach is about 3.6% higher than the optimal total cost. On Day 11, due to the size of the model (DA) (see Table 2), the problem could not even be read by the solver. Therefore, no solution is returned.

Finally, we present certain characteristics of the optimal solutions returned by the Hierarchical Approach on each of the 11 days. Table 5 presents, for each day, the number of demand points placing an order (n), (i) the total number of vehicles used, (ii) the number of different vehicle classes, (iii) the average number of different item types on each vehicle, (iv) the average percentage of used capacity in each vehicle, (v) the largest percentage of used capacity in any vehicle, (vi) the average number of demand points served by each vehicle, and (vii) the average number of vehicles serving each demand point. It is worth noticing that the number of different vehicle classes is usually considerably smaller than the number of retailers that placed an order, which implies that the problem, in general, has a nontrivial optimal solution. Another important observation is that, on average, each vehicle serves more than one demand point and each demand point is served by more than one vehicle, which illustrates that split deliveries indeed pay off. Finally, an examination of the columns (iv) and (v) reveals that the average utilization of each vehicle is considerably high. Furthermore, the fact that the highest utilization is 100% on several days indicates that the capacity constraint can be tight in the second stage problem (HA2). Therefore, the minimization of the excess capacity seems to be a meaningful objective function in (HA2). We remark that we observed similar characteristics with the solutions returned by the Direct Approach on each day except for Days 7 and 11. Recall that the Direct Approach returns a suboptimal solution on Day 7 and fails to compute a solution on Day 11.

Table 5
Characteristics of optimal solutions: (i) total number of vehicles used in transportation, (ii) number of different vehicle classes, (iii) average number of different item types on each vehicle, (iv) average percentage of used capacity in each vehicle, (v) the largest percentage of used capacity in any vehicle, (vi) average number of demand points served by each vehicle, and (vii) average number of vehicles serving each demand point.

Day	n	(i)	(ii)	(iii)	(iv)	(v)	(vi)	(vii)
1	14	14	9	13.71	84.67	99.69	1.50	1.50
2	11	15	11	12.60	87.38	100.00	1.13	1.55
3	12	13	8	14.46	93.41	99.94	1.54	1.67
4	17	17	11	14.82	95.74	99.88	1.59	1.59
5	10	7	6	15.43	81.45	94.24	1.57	1.10
6	19	15	12	16.40	88.42	99.90	1.60	1.26
7	23	19	12	17.16	92.55	99.86	1.68	1.39
8	10	18	8	10.78	87.06	100.00	1.17	2.10
9	7	10	6	12.60	83.23	99.94	1.10	1.57
10	8	10	7	16.70	97.12	100.00	1.40	1.75
11	27	22	14	17.05	91.07	100.00	1.59	1.30

Note that the feasibility restoration phase, as outlined in Theorem 3.2, has never been invoked on our data set since the sufficient condition of Lemma 3.1 was satisfied on each of the 11 days. Based on our limited data set, it is not possible to conclude whether this is a typical situation in real-life problems. Our computational experiments reveal that the Hierarchical Approach is especially well-suited for this particular data set. In any case, we first suggest the use of the Direct Approach on a given instance of the MDPFAC. If the Direct Approach fails to return an optimal solution, which seems to be the case especially for larger problem instances, then the Hierarchical Approach can be invoked, followed by the feasibility restoration phase, if necessary.

5. Concluding remarks

In this paper, we studied a multicommodity distribution problem under a special cost structure motivated by a real-life application. The direct formulation approach attempts to solve the MDPFAC directly using an integer programming model whereas the Hierarchical Approach consists of a partial demand aggregation scheme in the first stage followed by a second disaggregation stage. We analyzed the quality of solutions computed by the Hierarchical Approach. Our analysis was based on establishing several interesting links between our problem and the machine scheduling problem on parallel unrelated machines. Our computational experiments on a real-life data set revealed that the Hierarchical Approach significantly outperforms the direct formulation approach both in terms of the solution time and the quality of the solution, especially for larger instances.

After the original version of this manuscript was submitted, it was suggested by an anonymous reviewer that an instance of the multicommodity Split Delivery VRP (SDVRP) can be approximated by an instance of the MDPFAC. Similar to the location-based heuristic of Bramel and Simchi-Levi [4], the vehicle classes can be considered as "seeds" and overhead costs can be viewed as "insertion costs". It follows that our Hierarchical Approach can be used to compute an approximate solution of the multicommodity SDVRP.

In this paper, we analyzed an aggregation scheme in an attempt to solve large-scale instances of the MDPFAC. Future research directions include investigation of other approaches based on Lagrangian relaxation, column generation, and decomposition methods similar to the ones applied in the context of the concentrator location and the multiproduct capacitated facility location problems.

Acknowledgments

We are grateful to two anonymous referees and the Editor for insightful comments and suggestions, which considerably improved the presentation.

References

[1] Archetti C, Hertz A, Speranza M. A tabu search algorithm for the split delivery vehicle routing problem. Transportation Science 2006;40:64–73.
[2] Archetti C, Speranza MG. The split delivery vehicle routing problem: a survey. In: Golden B, Raghavan S, Wasil E, editors. The vehicle routing problem: latest

advances and new challenges. Operations research/computer science interfaces series, vol. 43. New York, NY, USA: Springer; 2008. p. 103–22.
[3] Archetti C, Speranza MG. Vehicle routing problems with split deliveries. International Transactions on Operational Research 2012;19:3–22.
[4] Bramel J, Simchi-Levi D. A location based heuristic for general routing problems. Operations Research 1995;43:649–60.
[5] Canel C, Khumawala BM, Law J, Loh A. An algorithm for the capacitated, multi-commodity multi-period facility location problem. Computers & Operations Research 2001;28:411–27.
[6] Chen S, Golden B, Wasil E. The split delivery vehicle routing problem: applications, algorithms, test problems, and computational results. Networks 2007;49(4):318–29.
[7] Drezner Z, editor. Facility location: a survey of applications and methods. New York, NY, USA: Springer; 1995.
[8] Drezner Z, Hamacher HW, editors. Facility location: applications and theory. Heidelberg, Germany: Springer; 2002.
[9] Dror M, Trudeau P. Savings by split delivery routing. Transportation Science 1989;23:141–5.
[10] Dror M, Trudeau P. Split delivery routing. Naval Research Logistics 1990;37: 383–402.
[11] Geoffrion AM, Graves GW. Multicommodity distribution system design by Benders decomposition. Management Science 1974;20:822–44.
[12] Hindi KS, Basta T. Computationally efficient solution of a multiproduct, two-stage distribution-location problem. The Journal of the Operational Research Society 1994;45:1316–23.
[13] Khachiyan LG. A polynomial time algorithm in linear programming. Soviet Mathematics Doklady 1979;20:191–4.
[14] Klincewicz JG. A large-scale distribution and location model. AT& T Technical Journal 1985;64:1705–30.
[15] Klincewicz JG. Hub location in backbone/tributary network design: a review. Location Science 1998;6:307–35.
[16] Klose A, Drexl A. Facility location models for distribution system design. European Journal of Operational Research 2005;162:4–29.
[17] Lee CY. The multiproduct capacitated facility location problem with a choice of facility type. Computers & Operations Research 1991;18:167–82.
[18] Lenstra JK, Shmoys DB, Tardos É. Approximation algorithms for scheduling unrelated parallel machines. Mathematical Programming 1990;46:259–71.
[19] Litvinchev I, Rangel S, Saucedo J. A Lagrangian bound for many-to-many assignment problems. Journal of Combinatorial Optimization 2010;19: 241–57.
[20] Mazzola JB, Neebe AW. Lagrangian-relaxation-based solution procedures for a multiproduct capacitated facility location problem with choice of facility type. European Journal of Operational Research 1999;115:285–99.
[21] Melo MT, Nickel S, Saldanha da Gama F. Dynamic multi-commodity capacitated facility location: a mathematical modeling framework for strategic supply chain planning. Computers & Operations Research 2006;33:181–208.
[22] Melo MT, Nickel S, Saldanha da Gama F. Facility location and supply chain management: a review. European Journal of Operational Research 2009;196: 401–12.
[23] Pirkul H, Narasimhan S, De P. Locating concentrators for primary and secondary coverage in a computer communications network. IEEE Transactions on Communications 1988;36:450–8.
[24] ReVelle CS, Eiselt HA. Location analysis: a synthesis and survey. European Journal of Operational Research 2005;165:1–19.
[25] ReVelle CS, Eiselt HA, Daskin MS. A bibliography for some fundamental problem categories in discrete location science. European Journal of Operational Research 2008;184:817–48.
[26] Shchepin EV, Vakhania N. An optimal rounding gives a better approximation for scheduling unrelated machines. Operations Research Letters 2005;33:127–33.
[27] Tang DT, Woo LS, Bahl LR. Optimization of teleprocessing networks with concentrators and multiconnected terminals. IEEE Transactions on Computers 1978;C-27:594–604.
[28] Yaman H. Concentration location in telecommunications networks. Combinatorial optimization. New York, NY, USA: Springer; 2005.